## Claims

**What is claimed is:**

1. A codevector trainer comprising:

an input data vector memory for storing a plurality of input data vectors therein;

a codevector memory for storing j codevectors therein;

i × j vector distance calculating units forming an array of i rows and j columns, the i × j vector distance calculating units being in data communication with the input memory and the codevector memory, the i × j vector distance calculating units for performing the steps of:

a) receiving one component of each of i input data vectors from the input data vector memory and one component of each of the j codevectors from the codevector memory, the components corresponding to a same dimension within the vectors;

b) determining i × j vector component distances corresponding to i × j combinations of the i input data vectors and the j codevectors;

c) summing the successive i × j vector component distances such that each of the successive i × j vector component distances is added in a respective vector distance calculating unit of the i × j vector distance calculating units in order to determine i × j scalar vector distances; and,

d) repeating the steps a), b) and c) for successive components;

e) accumulating the i input data vectors in j partitions, the j partitions corresponding to the j codevectors, the i input data vectors being accumulated in dependence upon their assignment to one of the j partitions;

and,

a control unit in data communication with the input data vector memory, the codevector memory, and the i × j vector distance calculating units for performing the steps of:

controlling provision of the components of the input data vectors and the codevectors to the respective vector distance calculating units;

determining a best match codevector for each of the i input data vectors in dependence upon the scalar vector distances;

assigning each of the i input data vectors to one of the j partitions in dependence upon the best match codevector; and,

updating the codevectors in dependence upon the accumulated input data vectors associated with j partitions.

2. A codevector trainer as defined in claim 1 wherein the IC is a FPGA.

3. A codevector trainer as defined in claim 1 wherein i × j vector component distances at a vector dimension are determined per clock cycle.

4. A codevector trainer as defined in claim 3 wherein the input data vectors comprise spectral vectors.

5. A codevector trainer comprising:

an input data vector memory for storing a plurality of input data vectors therein;

a codevector memory for storing N codevectors therein;

a vector distance calculator for determining a vectored form of a distance between an input data vector and a codevector provided thereto;

a scalar distance block for receiving the vectored form of the distance and for summing vector components of the distance in order to provide a scalar distance;

a minimal distance selector block for determining a minimum scalar distance between an input vector and the N codevectors;

N vector accumulators, each of the N vector accumulators for receiving the input data vectors associated with one of the N codevectors based upon the minimum scalar distance and for accumulating the received input data vectors;

a codebook update process block for updating the N codevectors based upon the accumulated input data vectors; and,

an exit criteria block for determining a termination of the codevector training process based upon the minimum scalar distances and a predetermined threshold.

6. A compression engine comprising:

an input port for receiving input data vectors;
an output port for providing codevectors and an index map;
volatile memory in data communication with the input port and the output port for storing the input data vectors, the codevectors and the index map;
a codevector trainer in data communication with the volatile memory for training codevectors in dependence upon the input data vectors using one of along vector components codevector training or across vector components codevector training; and,
a controller in communication with the codevector trainer and the volatile memory for executing in cooperation with the codevector trainer one of SAMVQ or HSOCVQ data compression in order to produce compressed data comprising the codevectors and the index map.

7. A compression engine as defined in claim 6 comprising a programming bus connected to a programming port, the codevector trainer, the controller, and the volatile memory for transmitting a control signal.

8. A compression engine as defined in claim 7 wherein the codevector trainer comprises hardware units for executing one of along vector components codevector training and across vector components codevector training in dependence upon a received control signal.

9. A compression engine as defined in claim 8 wherein the controller is capable of executing either one of a Successive Approximation Vector Quantization or a Hierarchical Cluster Vector Quantization.

10. A compression engine as defined in claim 9 wherein the data compression is performed within the compression engine without external communication during the compression process.

11. A compression engine as defined in claim 10 comprising a clock domain decoupled from a clock domain of a system environment with which it is in communication.

12. A compression engine as defined in claim 11 wherein the hardware components of the compression engine are accommodated within a single IC.

13. A compression engine as defined in claim 12 wherein the IC is a FPGA.

14. A compressor comprising:
a plurality of compression engines for simultaneously compressing a plurality of data subsets of a set of input data vectors and providing compressed data thereof using one of SAMVQ or HSOCVQ data compression;
an input port for receiving the set of input data vectors;
an output port for providing the compressed data;
a network switch in data communication with the input port, the output port, and the plurality of compression engines, the network switch for performing the steps of:
partitioning the set of input data vectors into the plurality of data subsets;
providing each of the plurality of data subsets to one of the plurality of compression engines; and,
transmitting the compressed data.

15. A compressor as defined in claim 14 comprising a programming bus connected to a programming port, the network switch, and the plurality of compression engines for transmitting a control signal.

16. A compressor as defined in claim 15 wherein each of the plurality of compression engines comprises hardware units for executing one of along vector components codevector training and across vector components codevector training in dependence upon a received control signal.

17. A compressor as defined in claim 16 wherein each of the plurality of compression engines is capable of executing one of a Successive Approximation Vector Quantization or a Hierarchical Cluster Vector Quantization in dependence upon a received control signal.

18. A compressor as defined in claim 17 wherein the data compression is performed within each of the plurality of compression engines without external communication during the compression process.

19. A compressor as defined in claim 18 wherein each of the plurality of compression engines comprises an independent clock domain.

20. A compressor as defined in claim 19 comprising an input buffer memory in data communication with the network switch.

21. A compressor as defined in claim 19 wherein hardware components of each of the plurality of compression engines and the network switch are accommodated each on a single IC connected to a PCB board.

22. A compressor as defined in claim 21 comprising a card to card connector for providing data and programming communication to a second PCB board.

23. A compressor as defined in claim 22 wherein the second PCB board comprises at least a compression engine.

24. A method for codevector training comprising the steps of:
a) receiving at an array of i rows and j columns of vector distance calculating units one component of each of i input data vectors and one component of each of j codevectors, the components corresponding to a same dimension within the vectors;
b) using the array of vector distance calculating units determining i × j vector component distances corresponding to i × j combinations of the i input data vectors and the j codevectors;
c) summing the successive i × j vector component distances such that each of the successive i × j vector component distances is added in a respective vector distance calculating unit of the i × j vector distance calculating units in order to determine i × j scalar vector distances;
d) repeating the steps a), b) and c) for successive components;
e) determining a best match codevector for each of the i input data vectors in dependence upon the scalar vector distances;
f) assigning each of the i input data vectors to one of j partitions in dependence upon the best match codevector, the j partitions corresponding to the j codevectors;

g) using the array of vector distance calculating units accumulating the i input data vectors in the j partitions in dependence upon their assignment; and,
h) updating the codevectors in dependence upon the accumulated input data vectors.

25. A method for codevector training as defined in claim 24 comprising the step of grouping a set of input data vectors into a plurality of pages of i input data vectors.

26. A method for codevector training as defined in claim 25 wherein steps a) to h) are repeated until each of the plurality of pages has been processed.

27. A method for codevector training as defined in claim 26 wherein the input data vectors comprise spectral vectors.

28. A method for codevector training comprising the steps of:
a) receiving at a vector distance calculator an input data vector of a plurality of input data vectors and one codevector of N codevectors;
b) using the vector distance calculator determining a vectored form of a distance between the input data vector and the codevector;
c) summing the vector components of the distance in order to provide a scalar distance;
d) repeating steps a) to c) for determining N scalar distances between the input data vector and the N codevectors, respectively;
e) determining a minimum scalar distance between the input data vector and the N codevectors;
f) providing the input data vector to one of N vector accumulators associated with the codevector having the minimum scalar distance to the input data vector;
g) repeating steps a) to f) for each of the plurality of input data vectors;
h) accumulating the respective input data vectors in each of the N vector accumulators;
i) updating the N codevectors based upon the accumulated input data vectors; and,
j) repeating steps a) to i) until the minimum scalar distances meet predetermined criteria.